arsatija / steam-reviews-ML

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

0 stars 0 forks 1 watching 1 Branch 0 Tags Activity

Public repository

.. 1 Branch 0 Tags Go to file t Go to file + Add file Code

arsatija Readme finished 0e8b5b2 · 28 minutes ago

img	Readme finished	28 minutes ago
notebooks	Readme finished	28 minutes ago
.gitignore	initial notebook and gitignore	3 hours ago
README.md	Readme finished	28 minutes ago

README

Steam Reviews Prediction

Business Understanding

The objective of the notebook is to be able to predict whether or not a game will be recommended based off the review. By analyzing reviews, the model identifies key words and phrases that lead to positive or negative recommendations. This can help game developers and marketers understand what aspects of a game are most appreciated by users when creating a sequel or a game in a specific genre.

Data Understanding

The data we will be looking at is a dataset acquired from Kaggle containing Steam reviews from 2017 and before.

> https://www.kaggle.com/datasets/andrewmvd/steam-reviews

The data will be downloaded and placed in the following path with the following name:

```
`/data/dataset.csv`
```

It should be about a 2.6 gb file when unzipped. A sneak peek of the data:

	app_id	app_name	review_text	review_score	review_votes
0	10	Counter-Strike	Ruined my life.	1	0

	app_id	app_name	review_text	review_score	review_votes
1	10	Counter-Strike	This will be more of a ''my experience with th...	1	1
2	10	Counter-Strike	This game saved my virginity.	1	0
3	10	Counter-Strike	• Do you like original games? • Do you like ga...	1	0
4	10	Counter-Strike	Easy to learn, hard to master.	1	1

We can see that the data has over 6 million rows with the following structure:

1. `app_id`
   - Game ID
2. `app_name`
   - Game Name
3. `review_text`
   - Review Content
4. `review_score`
   - 1 for recommended, -1 for not recommended
5. `review_votes`
   - number of votes for how helpful the review was

The rows that will be relevant for training the model will be `review_score` and `review_votes` as we just need the content of the review and whether or not it was a positive review.

Data Preparation

We need to remove the duplicate and NA rows which cuts the data down about 2 million entries. Once complete, we are going to be looking at only the two aforementioned rows in a 5% sample size. This sample size can be adjusted to affect accuracy of the model but also time complexity of the model. We settled with 5% for now to keep the model at a reasonable train time on a local machine. The next step would be to run the train test split with a 20% test size from that data. Next we will vectorize the data using the `TFID Vectorizer` from `scikit-learn` removing english stopwords and looking at unigrams to trigrams.

Modeling

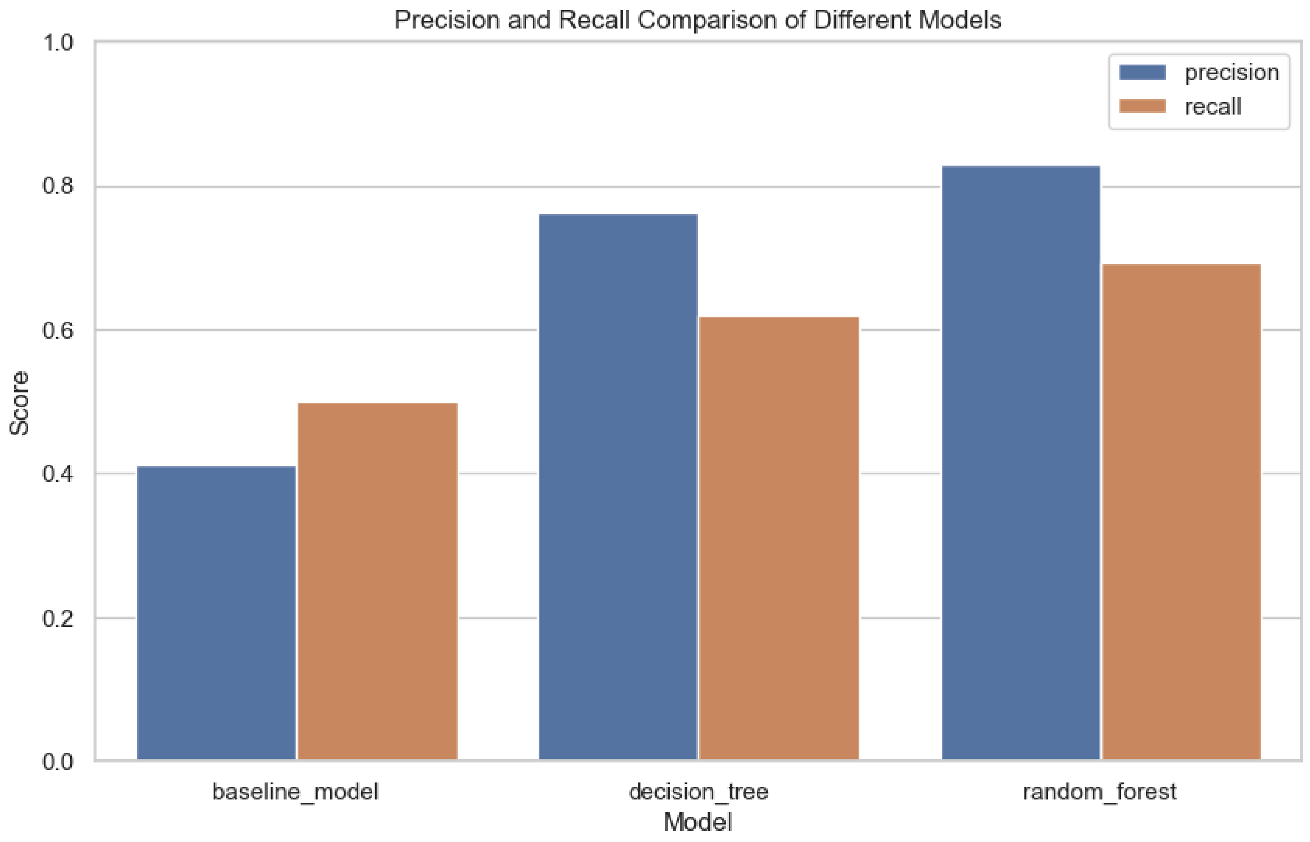
For this project we are going to create 3 models.

1. **Baseline Model:** Dummy Classifier Model
2. **First Model:** Decision Tree Classifier
3. **Improved Model:** Random Forest Classifier

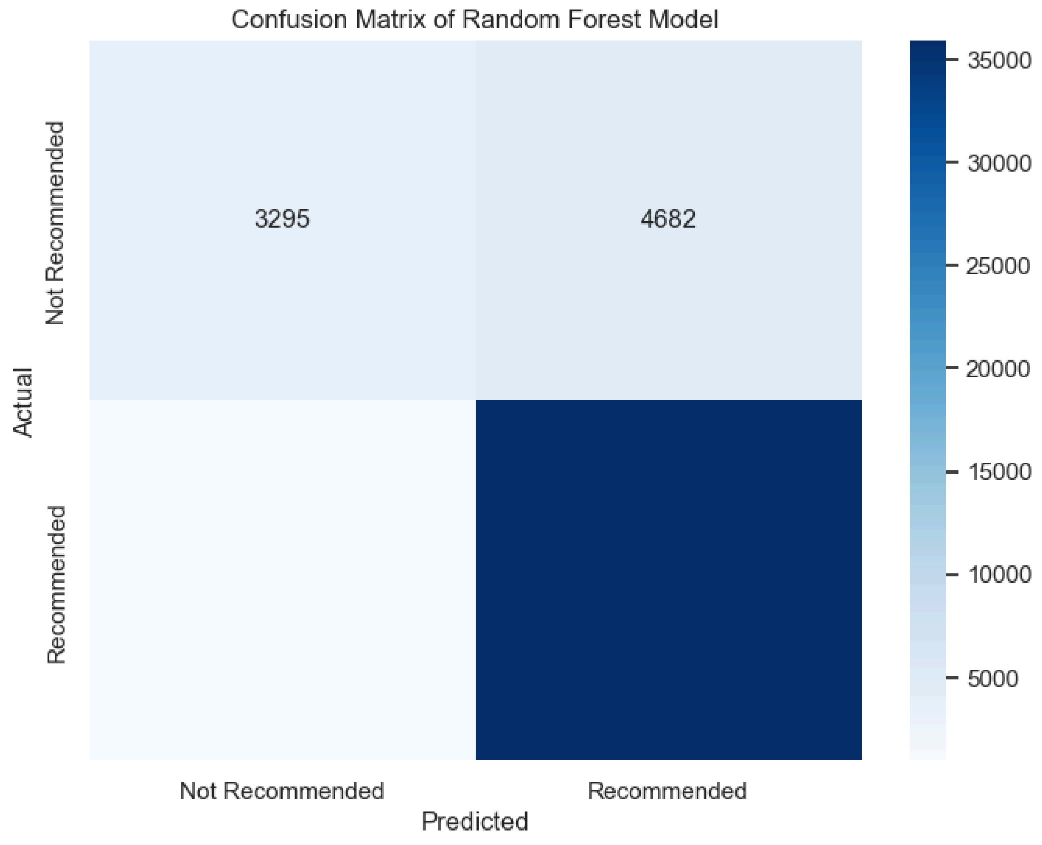
We will create a baseline model that should not be very good just to have a baseline to start comparing the other models to. We then start with a Decision Tree model with an okay performance that can definitely be improved with hypertuning with a GridSearch. However, upon further research I found that a Random Forest model may be better for this use case and it in fact was with the base parameters. We can further fine tune the Random Forest Classifier , given more time, by feeding it more data or messing with hyperparameters for this model as well.

Evaluation

I stored the precision and recall scores of each model in a dataframe to then represent in the following bar graph to help conclude which model performed the best. It was evidently the "Improved Model" of the **Random Forest Classifier**.

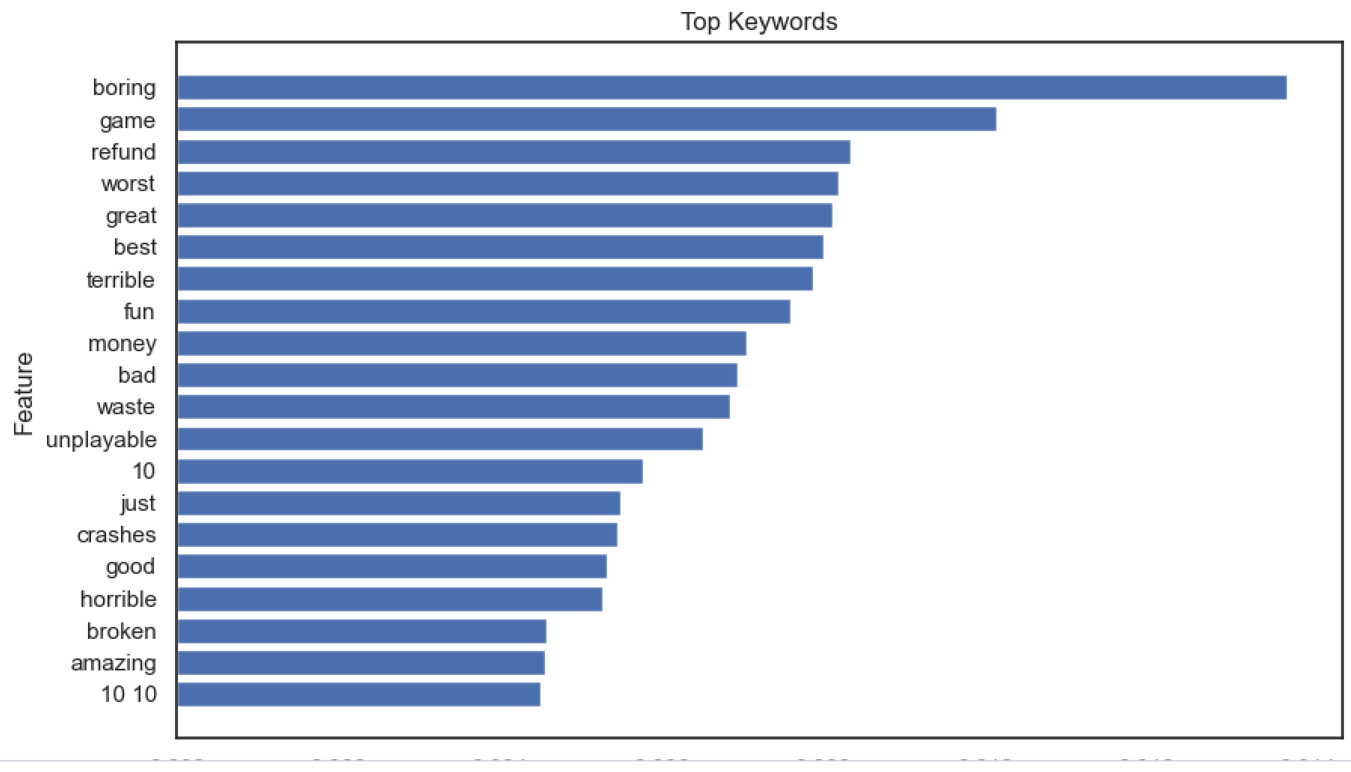


We can look at the confusion matrix for this model to get a better understanding of the precision of the model.



We can see a lot of overfitting however, this is much preferred to underfitting data in a situation like this. The crux of this project is to determine features that contribute to positive reviews rather than focusing on the negative features. There were almost no false positives which is really good for our intentions of using the model.

The purpose of this project is to find features that contribute to positive reviews, so let's look at the most important features as deemed by the model:



Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● **Jupyter Notebook** 100.0%